Introduction to Software Architecture (1)

Wendy Liu © 2003

(Acknowledgement: part of the content is contributed by Peter Kanareitsev)

Architect’s roles – not just technology

- Creating the “right” technical vision – aligned with organization’s business strategy
- Getting the organization to “buy into” this vision: executive sponsors, project managers, developers – requires a lot of political skill
- Leadership and communication – read Role of the Software Architect for more on this
- We will focus on technical aspects
Purpose of Software Architecture

[Jazayeri et al. 2000]
• a vehicle for communication
  – architectural views: functional, concurrency, code, development, physical view
  – architectural description: UML, ADLs
• a manifestation of earliest design decisions
  – quality attributes: performance, availability, reliability, etc.
  – architectural styles
• a reusable, transferable abstraction

Software Architecture

• Is a craft, not an engineering discipline, at this point in time (Shaw & Garlan)
• Is a creative, not a routine, activity
• No formal notion of “optimal” architecture, given a product domain
• No repeatable methodology to write down optimal architecture
• Formal training less valuable than experience, including some degree of learning from your own errors (Penny)
However…

- This does not mean you cannot reuse any of the vast experience of architects who came before you
- Can reuse traditional formats and modeling notations for capturing and communicating architectures
- Can reuse architectural styles – popular structures of components and connectors, each having a defined impact on functionality and quality attributes

Architectural Design Stages

1. Requirements
2. Architecturally Significant Requirements (ASRs)
3. Architectural Decisions
4. Architectural Description
5. Building Computational Infrastructure
6. Detail Implementation
Architecture blueprint should be:

- Self-motivating: include some rationale with your architectural decisions. Don’t leave the reader wondering why you made these choices.
- Relevantly biased: not all viewpoints are equally important for all systems – focus on the right aspects (e.g. for an AI system, knowledge base structure & reasoning mechanisms – logical view – deserves more detail than deployment view)
- Simple yet decisive (the hard part)
- Based on known architectural styles

Architectural Styles

- Like patterns in class design
- Common styles (Shaw & Garlan 1996):
  - Pipes and filters (e.g. Unix shell, data processing)
  - Implicit invocation (e.g. GUI’s: when an event is announced, any interested components may process it)
  - Layered functionality (e.g. network protocols, graphics rendering)
  - Repository / blackboard (e.g. online transaction processing, pattern recognition)
  - Interpreter (e.g. Java Virtual Machine)
More on Architectural Styles

• Attribute-based architectural styles (Klein & Kazman 1999):
  – Synchronization
  – Data indirection
    • Abstract data repository
    • Publish/Subscribe
  – Layering
  – Simplex
• Attempt to quantify the impact of styles on a system
  – e.g. authors show that Repository reduces the number of code changes in situations where data producers and consumers evolve while data schema stays frozen

Architectural Description Languages (ADLs)

• ACME
  – interchange
• Adage and Meta-H
  – avionics system
• Aesop and UniCon
  – real-time analysis
• Darwin
  – π-calculus
  – analysis of distributed message-passing
• Rapide
  – design simulation and analysis
• Wright
  – formal specification and analysis
• UML Approach
  – direct link to implementation
  – practitioners like it
  – poor vocabulary for AD
  – informal, less automated analysis
Next Tutorial

- We present
  - format of practical architectural design document
  - examples

References